

Assignment - 7
Deep Learning Image Classification with CNN

1. Follow the instruction below and then report how the performance changed.(apply all at once)

- Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
 - Dropout layer at 20%.
 - Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
 - Max Pool layer with size 2×2.
 - Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
 - Dropout layer at 20%.
 - Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
 - Max Pool layer with size 2×2.
 - Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
 - Dropout layer at 20%.
 - Convolutional layer, 128 feature maps with a size of 3×3 and a rectifier activation function.
 - Max Pool layer with size 2×2.
 - Flatten layer.
 - Dropout layer at 20%.
 - Fully connected layer with 1024 units and a rectifier activation function.
 - Dropout layer at 20%.
 - Fully connected layer with 512 units and a rectifier activation function.
 - Dropout layer at 20%.
 - Fully connected output layer with 10 units and a Softmax activation function
- Did the performance change?

Ans

- 1.Created the model with the given specification..
- 2.Compiled the model.
3. Fit the training data into the model.
4. Evaluated the model.

```
Home Page - Select or cr x | Untitled8 - Jupyter Noteb | x | Untitled1 - Jupyter Noteb | x | Untitled - Jupyter Notebo | x | image_classification.py - x | +
localhost:8888/notebooks/Untitled8.ipynb
jupyter Untitled8 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
In [14]: import numpy as np
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.constraints import maxnorm
from keras.utils import np_utils
from keras.optimizers import SGD

# Fix random seed for reproducibility
np.random.seed(7)

# Load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

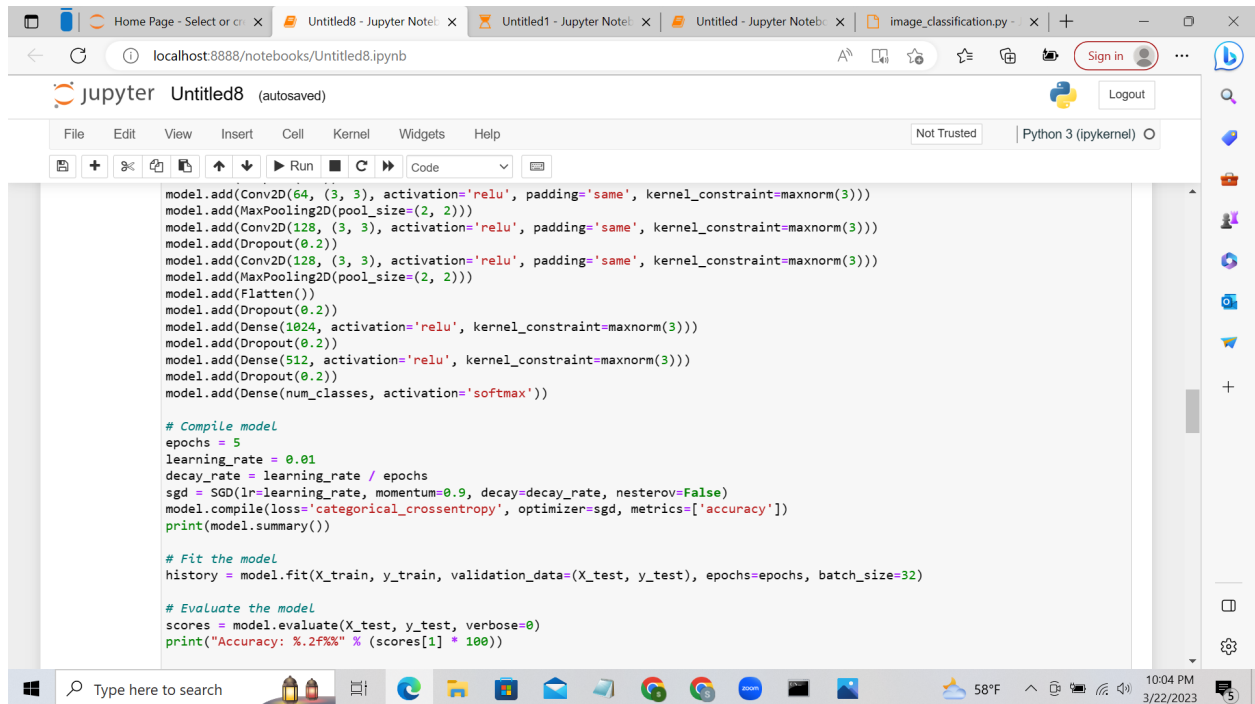
# Normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# One hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]

# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
```

```
Home Page - Select or cr x | Untitled8 - Jupyter Noteb | x | Untitled1 - Jupyter Noteb | x | Untitled - Jupyter Notebo | x | image_classification.py - x | +
localhost:8888/notebooks/Untitled8.ipynb
jupyter Untitled8 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
# Create the model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 5
learning_rate = 0.01
decay_rate = learning_rate / epochs
sgd = SGD(lr=learning_rate, momentum=0.9, decay=decay_rate, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
```



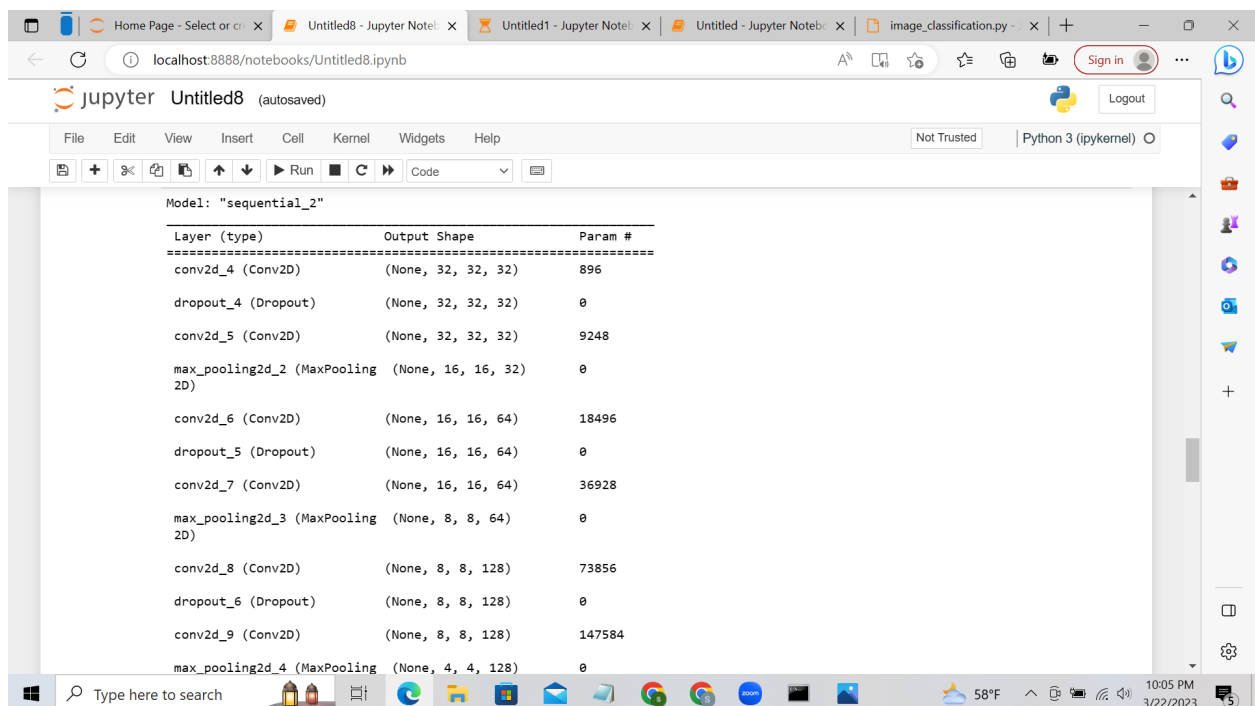
```
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 5
learning_rate = 0.01
decay_rate = learning_rate / epochs
sgd = SGD(lr=learning_rate, momentum=0.9, decay=decay_rate, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
print(model.summary())

# Fit the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)

# Evaluate the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1] * 100))
```

Output for the above code is



```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 32, 32, 32)	896
dropout_4 (Dropout)	(None, 32, 32, 32)	0
conv2d_5 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_2 (MaxPooling 2D)	(None, 16, 16, 32)	0
conv2d_6 (Conv2D)	(None, 16, 16, 64)	18496
dropout_5 (Dropout)	(None, 16, 16, 64)	0
conv2d_7 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_3 (MaxPooling 2D)	(None, 8, 8, 64)	0
conv2d_8 (Conv2D)	(None, 8, 8, 128)	73856
dropout_6 (Dropout)	(None, 8, 8, 128)	0
conv2d_9 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_4 (MaxPooling 2D)	(None, 4, 4, 128)	0

Home Page - Select or cr X Untitled8 - Jupyter Note Untitled1 - Jupyter Note Untitled - Jupyter Note image_classification.py X +

localhost:8888/notebooks/Untitled8.ipynb

Sign in Logout

Jupyter Untitled8 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
max_pooling2d_4 (MaxPooling (None, 4, 4, 128) 0
2D)

flatten_2 (Flatten) (None, 2048) 0

dropout_7 (Dropout) (None, 2048) 0

dense_4 (Dense) (None, 1024) 2098176

dropout_8 (Dropout) (None, 1024) 0

dense_5 (Dense) (None, 512) 524800

dropout_9 (Dropout) (None, 512) 0

dense_6 (Dense) (None, 10) 5130

=====
Total params: 2,915,114
Trainable params: 2,915,114
Non-trainable params: 0

None
Epoch 1/5
1563/1563 [=====] - 15s 9ms/step - loss: 1.9322 - accuracy: 0.2796 - val_loss: 1.6108 - val_accuracy:
0.4168
Epoch 2/5
1563/1563 [=====] - 13s 9ms/step - loss: 1.5375 - accuracy: 0.4379 - val_loss: 1.4261 - val_accuracy:
0.4795
```

Type here to search 58°F 10:07 PM 3/22/2023

Home Page - Select or cr X Untitled8 - Jupyter Note Untitled1 - Jupyter Note Untitled - Jupyter Note image_classification.py X +

localhost:8888/notebooks/Untitled8.ipynb

Sign in Logout

Jupyter Untitled8 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
dense_5 (Dense) (None, 512) 524800

dropout_9 (Dropout) (None, 512) 0

dense_6 (Dense) (None, 10) 5130

=====
Total params: 2,915,114
Trainable params: 2,915,114
Non-trainable params: 0

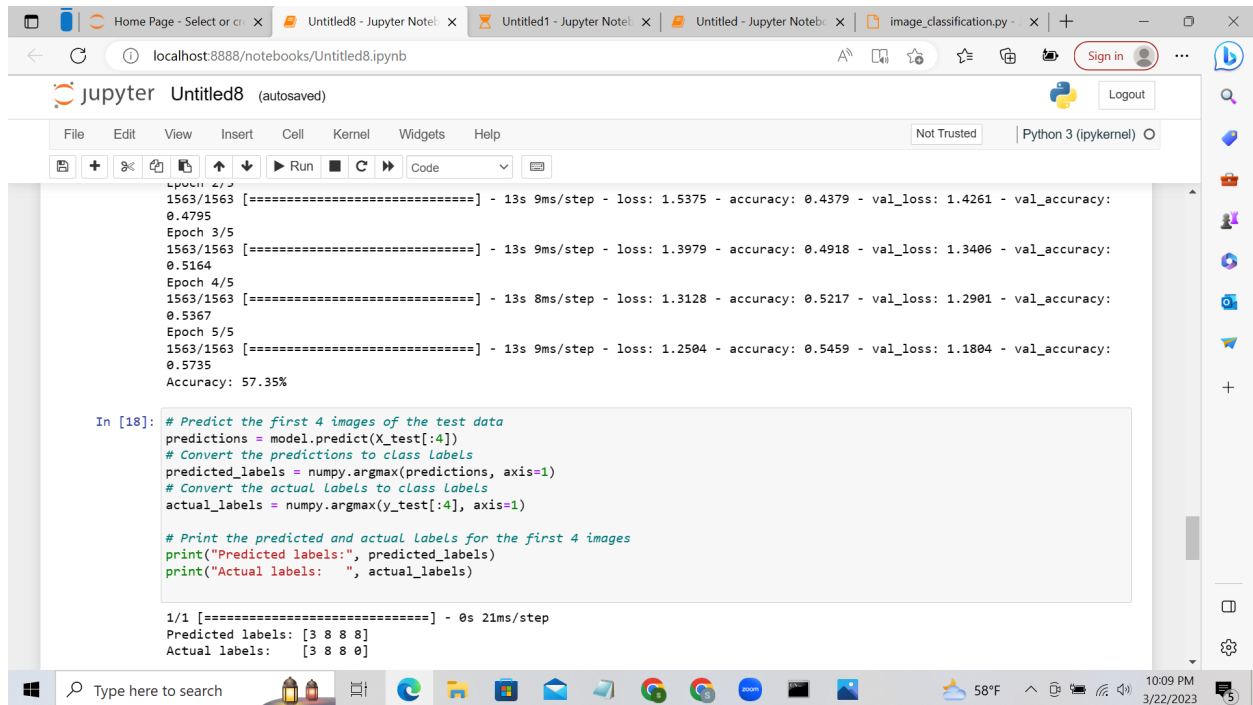
None
Epoch 1/5
1563/1563 [=====] - 15s 9ms/step - loss: 1.9322 - accuracy: 0.2796 - val_loss: 1.6108 - val_accuracy:
0.4168
Epoch 2/5
1563/1563 [=====] - 13s 9ms/step - loss: 1.5375 - accuracy: 0.4379 - val_loss: 1.4261 - val_accuracy:
0.4795
Epoch 3/5
1563/1563 [=====] - 13s 9ms/step - loss: 1.3979 - accuracy: 0.4918 - val_loss: 1.3406 - val_accuracy:
0.5164
Epoch 4/5
1563/1563 [=====] - 13s 8ms/step - loss: 1.3128 - accuracy: 0.5217 - val_loss: 1.2901 - val_accuracy:
0.5367
Epoch 5/5
1563/1563 [=====] - 13s 9ms/step - loss: 1.2504 - accuracy: 0.5459 - val_loss: 1.1804 - val_accuracy:
0.5735
Accuracy: 57.35%
```

Type here to search 58°F 10:08 PM 3/22/2023

2. Predict the first 4 images of the test data using the above model. Then, compare with the actual label for those 4 images to check whether or not the model has predicted correctly.

Ans

1. Predicted the first 4 images of the test data using the above model.
2. Converted the predictions to class labels
3. Converted the actual labels to class labels.
4. Printed the Predicted labels and actual labels.



The screenshot shows a Jupyter Notebook titled 'Untitled8 (autosaved)' running on a local host. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a code editor. The code editor contains the following text:

```
Epoch 4/5  
1563/1563 [=====] - 13s 9ms/step - loss: 1.5375 - accuracy: 0.4379 - val_loss: 1.4261 - val_accuracy:  
0.4795  
Epoch 3/5  
1563/1563 [=====] - 13s 9ms/step - loss: 1.3979 - accuracy: 0.4918 - val_loss: 1.3406 - val_accuracy:  
0.5164  
Epoch 4/5  
1563/1563 [=====] - 13s 8ms/step - loss: 1.3128 - accuracy: 0.5217 - val_loss: 1.2901 - val_accuracy:  
0.5367  
Epoch 5/5  
1563/1563 [=====] - 13s 9ms/step - loss: 1.2504 - accuracy: 0.5459 - val_loss: 1.1804 - val_accuracy:  
0.5735  
Accuracy: 57.35%
```

Below the training output, there is a code cell with the following code:

```
In [18]: # Predict the first 4 images of the test data  
predictions = model.predict(X_test[:4])  
# Convert the predictions to class labels  
predicted_labels = numpy.argmax(predictions, axis=1)  
# Convert the actual labels to class labels  
actual_labels = numpy.argmax(y_test[:4], axis=1)  
  
# Print the predicted and actual labels for the first 4 images  
print("Predicted labels:", predicted_labels)  
print("Actual labels:  ", actual_labels)
```

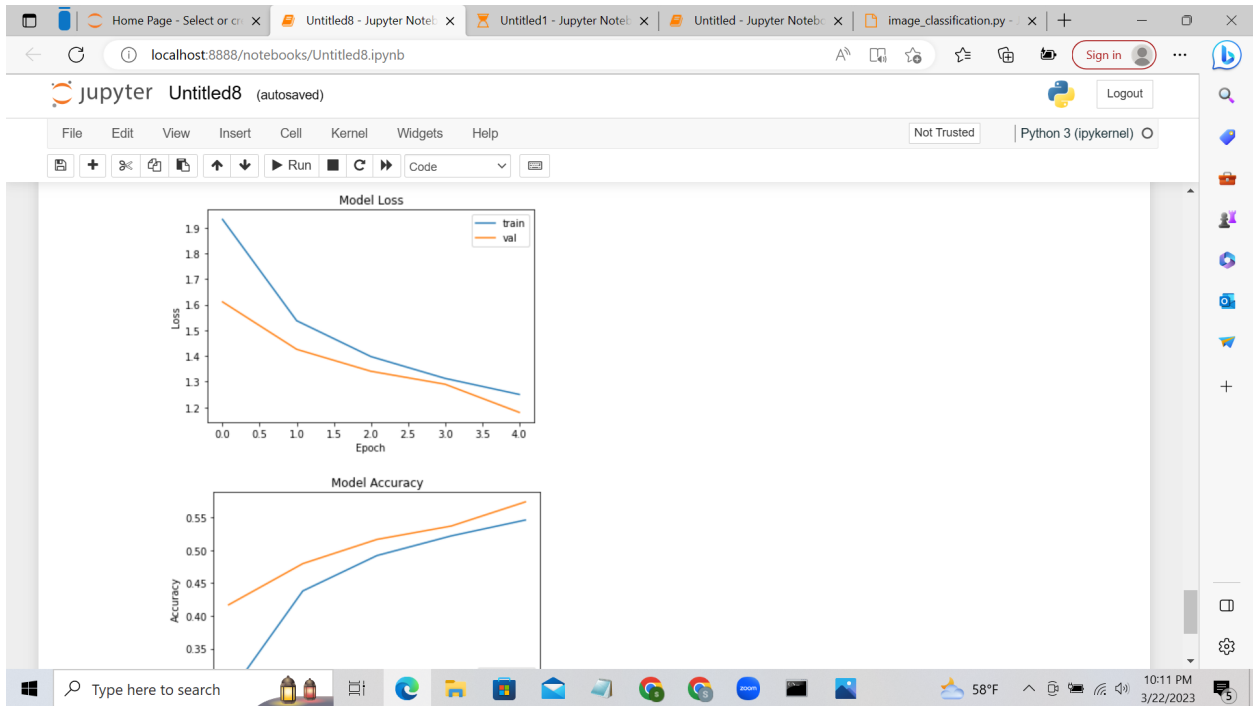
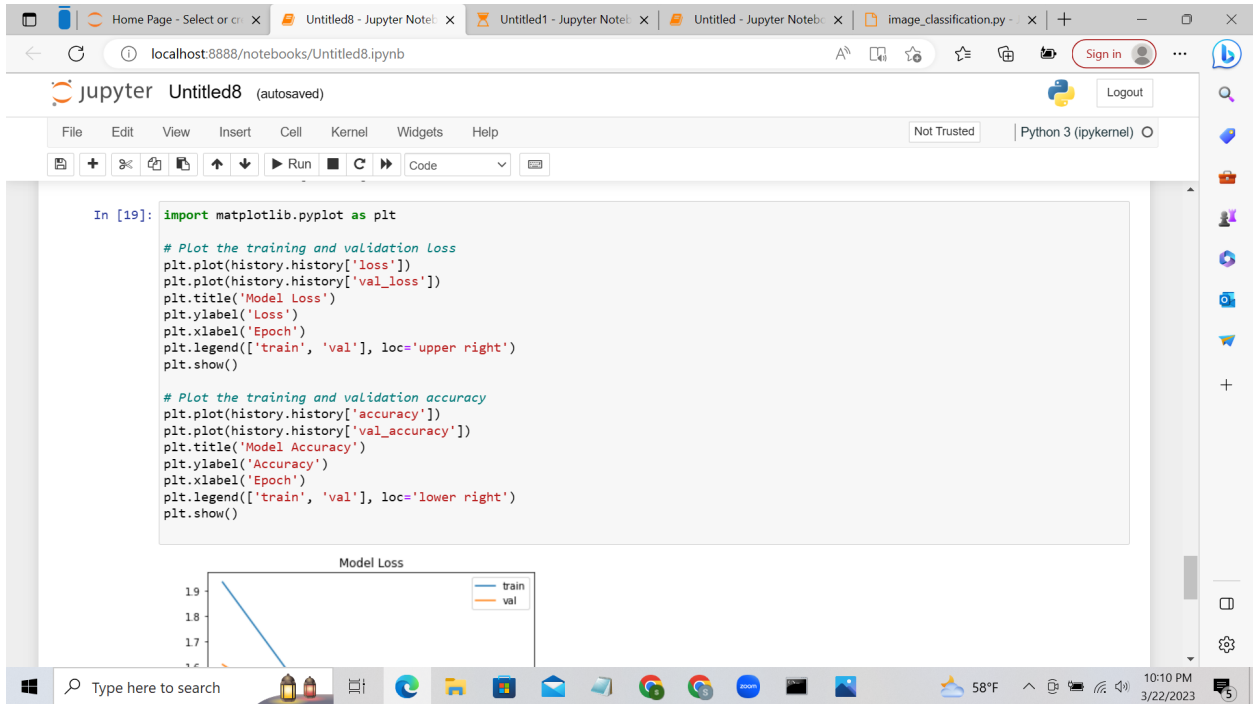
The output of the code cell shows:

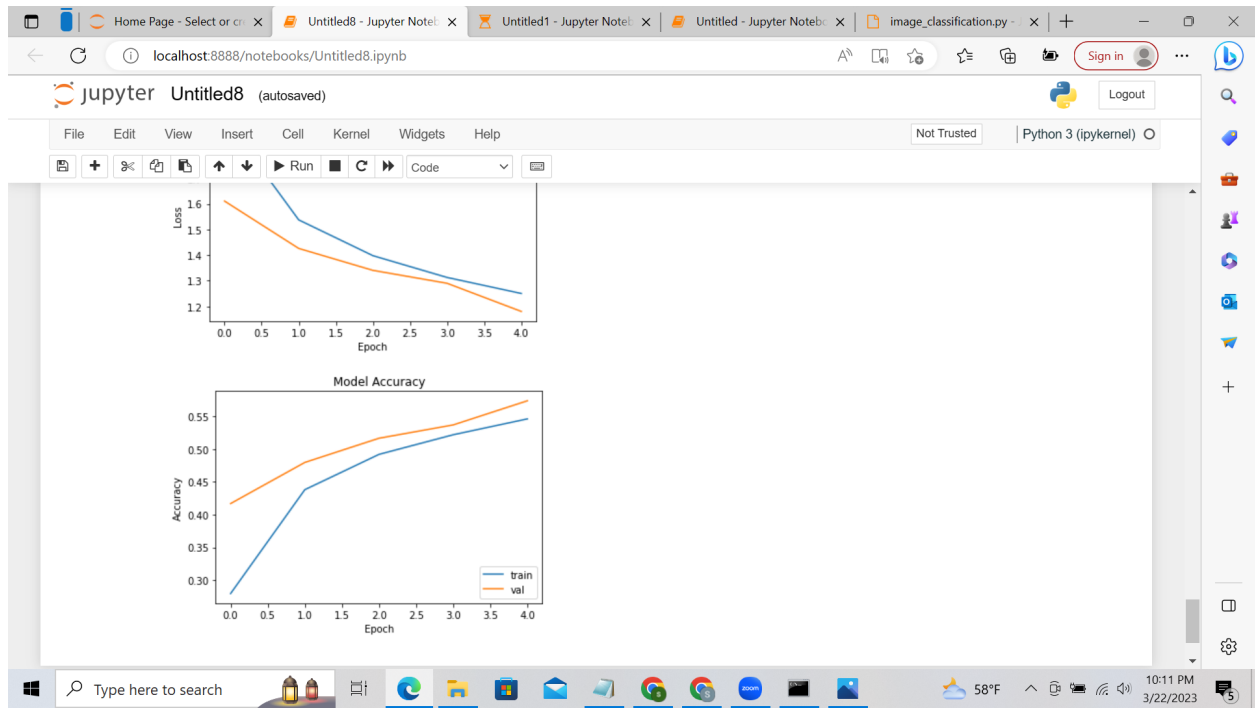
```
1/1 [=====] - 0s 21ms/step  
Predicted labels: [3 8 8 8]  
Actual labels:   [3 8 8 0]
```

3. Visualize Loss and Accuracy using the history object

Ans:

1. Visualized the Loss and Accuracy using the history object.





Git repo link <https://github.com/ShruthiVallapReddy/NNDL-Assignment-7.git>